Modelo de red neuronal digital para predicción climática

Tosini, Marcelo Alejandro* - Acosta, Gerardo** - Boemo, Eduardo Ivan***

* Grupo de Investigación en Computación Aplicada (INCA) - Fac. de ciencias Exactas - Universidad Nacional del Centro de la Prov. de Bs. As. (UNCPBA) – Campus universitario, Paraje Arroyo Seco s/n, Tandil (7000) – TE: +54 2293 432466
Email: mtosini@exa.unicen.edu.ar

** CONICET- Grupo ADQDAT – Dept. de Ing. Electromecánica – Fac. de Ingeniería - Universidad Nacional del Centro de la Prov. de Bs. As. (UNCPBA) – Av. Del Valle 5737, Olavarría (7400) – TE: +54 2284 451055/6, Fax: +54 2284 450628
Email: ggacosta@fiog.fio.unicen.edu.ar

*** Escuela Técnica Superior de Informática – Universidad Autónoma de Madrid – Ctra. Colmenar Km. 15, Madrid, España
Email: eib@ii.uam.es

Resumen

Se presenta en este artículo el desarrollo de una red neuronal artificial (RNA), implementada en un circuito digital, para realizar predicciones sobre variables climáticas en un medioambiente acotado con microclima propio. Estas variables (temperatura, humedad del suelo, ventilación,…) se desean mantener bajo control, y para ello es sumamente ventajoso disponer de un módulo capaz de predecir su evolución en un horizonte temporal tan amplio como sea posible. De este modo, la RNA llevará adelante la tarea de predicción dentro de un sistema mayor, basado en conocimiento, dedicado al control y supervisión de un invernadero.
Se propone una arquitectura para el sistema digital parametrizable y programable por el diseñador, así como un detalle de la metodología de diseño de la arquitectura y de la programación de la misma para distintas topologías de redes neuronales.
Se pretende, por último, un diseño final de costo accesible y altamente modificable, usando FPGAs como tecnología de base para su materialización.

Palabras Clave

Redes neuronales digitales, Sistemas digitales, automatización de plantas, inteligencia artificial.

0. Introducción

El prototipo experimental para la obtención de datos de prueba es un invernadero de 6x50m. en el cual, en esta primera etapa, sólo se obtienen datos referentes a temperatura interior y exterior y humedad del suelo. La medida de temperatura global se obtiene a partir de un arreglo de termistores distribuidos dentro del invernadero. Esta medición se ingresa en un bus serie conectado a una PC. Para la adquisición de datos referentes a la humedad del suelo se entierran cuatro sensores distribuidos equidistantemente en el área interior del invernadero. Además se incorporan al bus mediciones provenientes de una microestación meteorológica exterior que provee la radiación solar, velocidad y dirección del viento. Las acciones de control se realizan a través de un sistema de actuadores que abren o cierran linealmente una ventana de tipo zenith y una válvula automática de riego. La estrategia de control no es siempre la misma y varía según el tipo de cultivo, la estación del año y la etapa y estado del día.

Figura 1: prototipo del Invernadero y distribución de los sensores de temperatura (trazo punteado) y humedad (trazo de rayas)

A fin de supervisar y coordinar la serie de procesos involucrados en el funcionamiento de este prototipo de invernadero se está desarrollando un SBC, a partir de una taxonomía de tareas orientadas al análisis, comparación y desarrollo de este tipo de sistemas [Acosta et al (2000)]. Según la taxonomía propuesta, se identifican ocho grandes tareas de supervisión, a citar: Monitoreo, Identificación, Diagnosis, Prognosis, Modo de Operación, Planeamiento, Interface hombre-máquina (HMI) y Validación de datos. De estas tareas, la Prognosis es ejecutada por la RND, en una tarjeta independiente del SBC soportado en PC (figura 2), conectable directamente al bus serie. La RND provee de este modo conocimiento al SBC sobre la evolución de algunas variables climáticas vitales.

Concretamente, se busca implementar un modelo neuronal que, basado en cierto historial de la temperatura interior y la humedad del suelo, sea capaz de predecir sus tendencias de alguno de dichos parámetros en un futuro cercano (normalmente en el siguiente período de recopilación de datos).

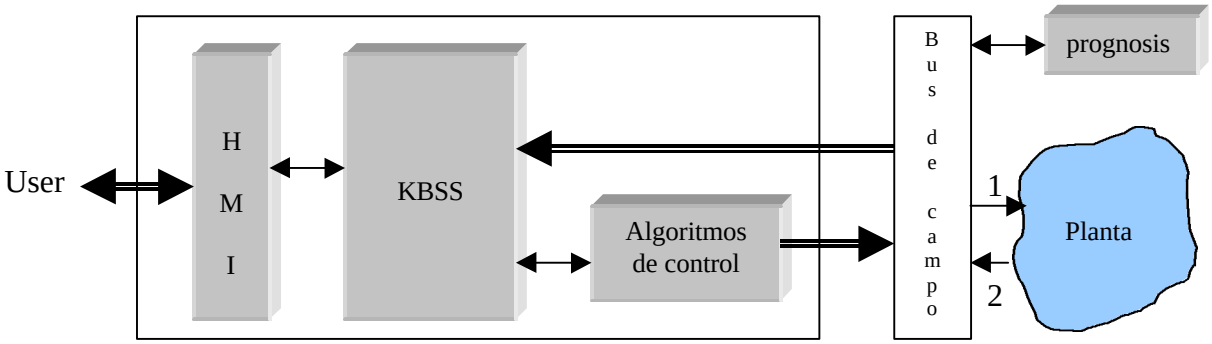


figura 2: Esquema simplificado del sistema PC – Bus – RNA Board – Planta
(1: actuadores; 2: sensores)

1. Redes neuronales aplicadas a la predicción climática

características generales

Existen distintas arquitecturas de redes neuronales capaces de aprender características evolutivas de series temporales a fin de predecir estados futuros de estas series en base a información presente y pasada de las mismas. El perceptrón multicapa (PMC) es frecuentemente usado para la predicción temporal de series. El vector de entrada de un PMC consistirá, entonces, de cierta cantidad *p* de muestras pasadas de la serie temporal en la forma $\boldsymbol{x}=[x(n-1), x(n-2),.., x(n-p)]$, siendo la salida de la red el valor $y(n) = x(n)$[Koskela, (1996)]. Su desventaja principal es que, debido a que la relación entre entradas y salidas es estática, sólo puede aplicarse a series temporales estacionarias. Una segunda alternativa consiste en el uso algún tipo de memoria a fin de almacenar información temporal relevante. Esta es la aproximación usada en las redes Elman y en las redes tipo FIR (Finite Impulse Response) [Koskela, (1996)][Corchado et al, (1998)]. Las primeras usan realimentación positiva de las salidas hacia las entradas en unidades de memoria denominadas ´de contexto´ a fin de incorporar en el aprendizaje comportamientos pasados de la red. En las redes FIR, por otro lado, los pesos de las sinapsis son reemplazados por filtros FIR de manera que estas redes se comportan como una extensión vectorial y temporal de los MLP.
Si en la aplicación a desarrollar el horizonte de predicción es lo suficientemente pequeño las series temporales pueden considerarse estacionarias. En base a lo anterior, y con el fin de obtener una arquitectura simple para la red neuronal a implementar, se optó por la utilización de una topología basada en el perceptrón multicapa con funciones de activación a la salida de las neuronas ocultas – Radix Basis Function, RBF. A partir de esta topología de red se implementó un modelo con realimentación de las salidas con el siguiente vector de regresión: [Hagan, (1994)]

$$\mathrm{j}\ (t) = [y(t-1), \ldots, y(t-n_i), u(t-n_k), \ldots, u(t-n_b-n_k+1)]$$

y siendo la función de predicción:

$$y(t|\mathrm{q}) = g(\mathrm{j}\ (t), \mathrm{q})$$

donde θ es el vector de pesos sinápticos y *g* es la función realizada por la red neuronal.

2. Arquitectura de la red neuronal digital propuesta

La arquitectura desarrollada se basa en dos características esenciales. Por un lado, se implementa una ruta de datos altamente utilizable a través del uso de componentes segmentados (multiplicador, sumador, fila de datos/resultados) que permiten obtener un alto rendimiento del circuito de datos,

con el consecuente incremento de velocidad. Por otro lado, la posibilidad de microprogramar la ruta de datos permite gran flexibilidad a la hora de implementar distintas alternativas de redes neuronales: feedforward, cíclicas, acíclicas, etc. La arquitectura general del circuito se muestra a continuación (figura 3) [Tosini, (2000)]

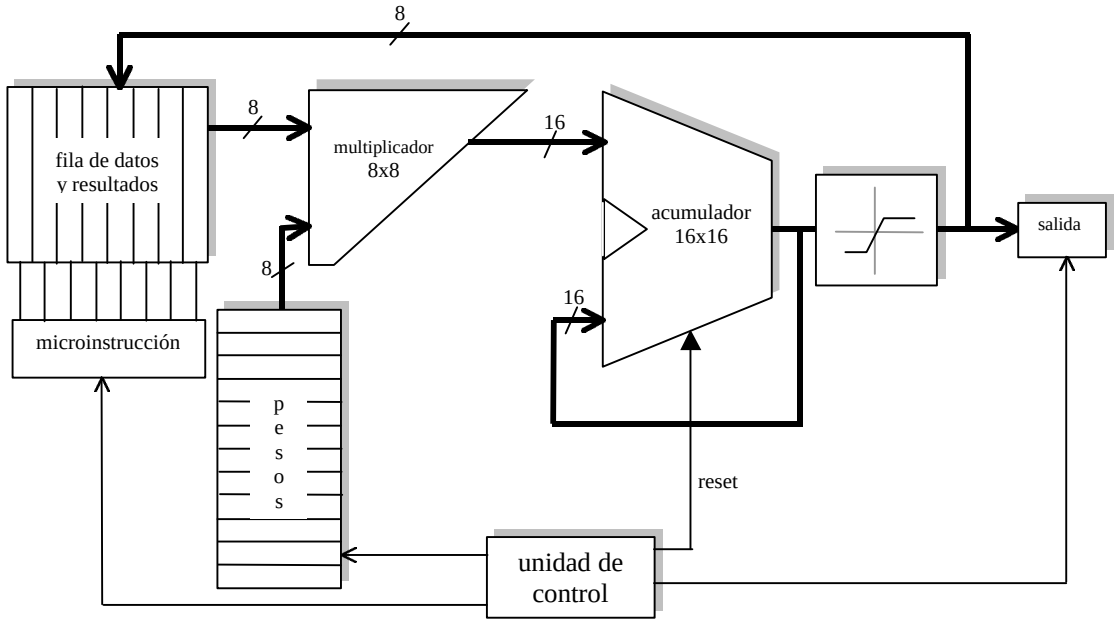


Figura 3: arquitectura general de la red neuronal segmentada y microprogramada

En el circuito diseñado se utilizó un multiplicador de dos entradas de 8 bits y una salida de 16 bits, con un total de 360 registros y 64 multiplicadores básicos de 1 bit. Esta configuración elegida de 8 bits para los datos internos es fácilmente reconfigurable, aunque se advierte rápidamente que datos de más bits encarecen la arquitectura propuesta en un factor cuadrático respecto de los recursos necesarios para implementar el multiplicador.
La ruta de datos así construida permite obtener un nuevo resultado en cada ciclo de reloj, siendo la latencia para cada cálculo de 32 ciclos.
La salida final del acumulador se aplica a la entrada del circuito de cálculo de la función de activación aunque, debido a las características del mismo, sólo se usan 9 de los 16 bits que aporta el conjunto sumador [A_6..A_{14}]. El bit 15 también se utiliza en la función de activación para determinar la pertenencia del valor de salida al conjunto [0..127] ó [128..255]. El resto de los bits [A_0..A_5] solamente se usan en la etapa de acumulación a efectos de mantener la precisión del cálculo.

Función de activación

En las implementaciones software de muchos tipos de redes neuronales la función de activación realiza usualmente una aproximación sigmoide de su entrada según la ecuación

$$\frac{1}{1+e^{-x}} \qquad [1]$$

Esta función es inconveniente en vista de materializaciones en hardware de redes neuronales debido a los altos costos de mantener una tabla de 'lookup' para la función exponencial sumado al costo adicional de una operación de división.
Una alternativa bastante difundida permite calcular la función sigmoide a partir de la siguiente aproximación [Nordström, (1982)]

$$\frac{1}{2}\left(\frac{x}{1+|x|}+1\right) \qquad [2]$$

Esta aproximación, si bien más simple, aún requiere circuitos necesarios para cálculo del módulo y para la división.
La alternativa presentada en este trabajo se basa en obtener una aproximación de la función sigmoide [1] con varias funciones polinomiales aproximadas a determinadas potencias de 2 (piecewise lineal - PWL) [Myers, (1989)][Alippi, (1991)].
Para este caso en particular, se desarrolló un circuito para la función de activación cuya entrada es un valor entero de 16 bits [-32768..32767] y su salida es un entero con signo de 8 bits [-1128..127].
La tabla siguiente muestra los 9 rangos de entrada utilizados con sus respectivos rangos de salida.

Entrada	**Salida**
0..1023	0..31
1024..2047	32..63
2048..4095	64..95
4096..5119	96..111
5120..6143	112..115
6144..7167	116..119
7168..8191	120..123
8192..12287	124..127
12288..32767	127

[3]

Tabla 1: Aproximación con funciones polinomiales aproximadas a potencias de 2

Los valores complementados de la entrada [-32768..0] darán los correspondientes valores de salida en el rango [-128..0].
La figura 4 muestra las respuestas correspondientes para la función sigmoide [1] y las dos aproximaciones [2] y [3].
Se observa que la utilización de funciones polinomiales [3] se aproxima más exactamente a la función sigmoide original [1] que la ecuación [2].

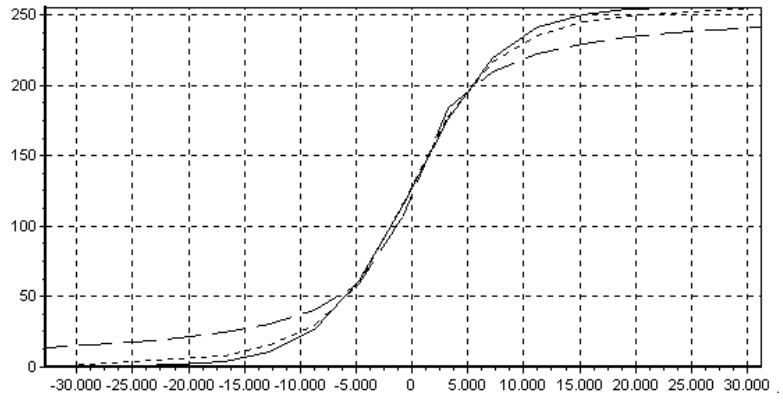


Figura 4: Generación de las tres funciones de activación. Función sigmoide con línea completa, aproximación por módulo de x [2] con trazos largos y generación a partir de tabla [3] con puntos.

La función de activación estudiada se implementó como un módulo en lenguaje de descripción de hardware VHDL, de manera que reemplazando dicho módulo en la ruta de datos se cambia fácilmente la función de salida de la red neuronal.

3. Programación de la red neuronal

La programación de la arquitectura propuesta se basa, por un lado, en la definición de una serie de parámetros y , por otro lado, en la creación de un microprograma que controla la evolución de los cálculos a realizar para resolver la red neuronal, objeto del diseño en particular.

Definición de parámetros

El primer parámetro es el número de celdas de la fila de datos/resultados (*l*). El mismo es equivalente a la suma máxima del número neuronas presentes en dos niveles contiguos menos 1, ya que dicha fila debe, en todo momento, poder almacenar las salidas del nivel ya calculado (datos) más los resultados de calcular cada neurona del nivel actual.
El otro parámetro es el tamaño de la memoria de pesos, la cual debe tener capacidad para almacenar todos los pesos de la red neuronal bajo diseño.

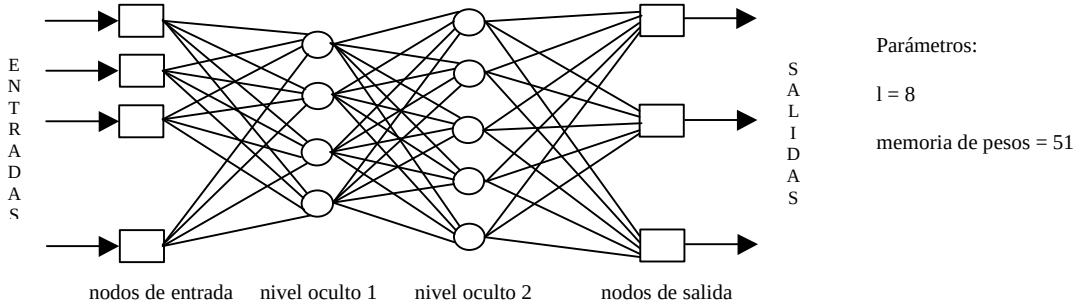


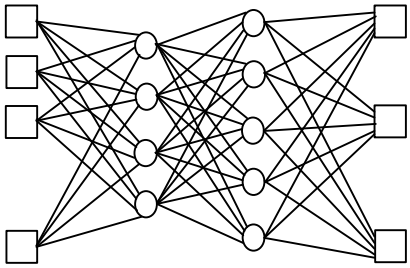
Figura 5: Red neuronal feedforward clásica con sus parámetros correspondientes

Creación del microprograma

Partiendo de una descripción de la red neuronal en varios niveles de detalle, se desarrolló una herramienta de software (ensamblador) que analiza las relaciones de precedencia de los cálculos a realizar para la red dada.
En el nivel más abstracto se da una descripción genérica de un tipo de red específico, siendo los tipos soportados hasta el momento los siguientes: red feedforward y Hopfield sincrónica y asincrónica [Gurney, (1995)].

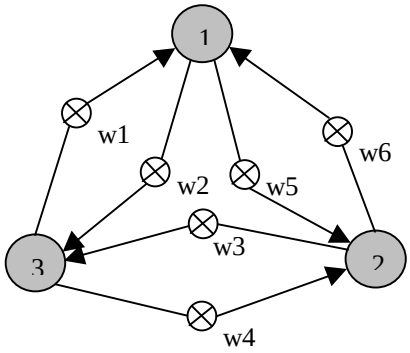
Una red feedforward genérica se define

Net <*name*> **Description**
    type feedforward
    Input : 4
    Output : 3
    Hidden : 4 , 5
End <*name*>



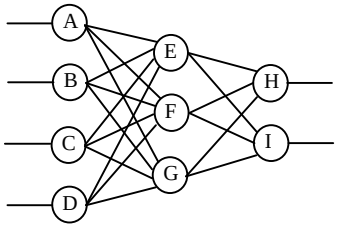
Una red Hopfield se define

Net <*name*> **Description**
type sync Hopfield
Nodes : 3
End <*name*>



La característica de sincrónica o asincrónica de las redes Hopfield permite al software realizar el microprograma correspondiente teniendo en cuenta que todas las neuronas se actualicen en conjunto en cada ciclo de la red o en forma secuencial.
Esta descripción de alto nivel sólo permite generar redes simples y altamente regulares, aunque no permite describir detalles específicos de la interacción entre neuronas en un diseño en particular.
Un nivel mas bajo de especificación obliga a expresar con mayor detalle las relaciones entre neuronas a partir de ecuaciones que manifiestan explícitamente las dependencias.
De esta forma se puede representar cualquier tipo de conexionado de las neuronas de la red así como el número de entradas de cada una de ellas.

Net <*name*> **Description**
type Feedforward
Nodes: A, B, C, D, E, F, G, H, I
Weights: w1..w18
Input Nodes: A, B, C, D
Output Nodes : H, I
Relation
E = A * w1 + B * w2 + C * w3 + D * w4
F = A * w5 + B * w6 + C * w7 + D * w8
G = A * w9 + B * w10 + C * w11 + D * w12
H = E * w13 + F * w14 + G * w15
I = E * w16 + F * w17 + G * w18
End Relation
End <*name*>



A partir de esta descripción, el ensamblador puede establecer los parámetros comentados en el apartado anterior, de manera que *l* será igual a 6 y la memoria de pesos será igual a 18 celdas.
La generación del microprograma se basa en las relaciones expresadas, las cuales se mapean en un conjunto de microinstrucciones que la unidad de control del circuito cargará sucesivamente en el registro de microinstrución de la fila de datos/resultados. Dicho registro tiene 2**l* bits donde cada par de bits codifica una de las cuatro operaciones posibles para cada celda de la fila :

rotación (C)	la celda en cuestión recibe el valor que sale de la fila.
no operación (N)	la celda en cuestión no se modifica.
desplazamiento (S)	la celda en cuestión recibe el valor de la anterior (izquierda).
carga de dato (L)	la celda en cuestión se carga con el valor proveniente de la función de activación o de la entrada de datos del circuito.

El microprograma resultante de la red mostrada anteriormente será el siguiente (sólo se muestra un extracto):

ciclo	microinstrucción	estado de la fila	operación	comentarios
0	NNNNNL	_ _ _ _ _ A	_	*carga de datos iniciales*
1	NNNNLN	_ _ _ _ B A	_	
2	NNNLNN	_ _ _ C B A	_	
3	NNLNNN	_ _ D C B A	_	
4	NNCSSS	_ _ A D C B	Acc = A * w1	*A recircula*
5	NNCSSS	_ _ B A D C	Acc += B * w2	*B recircula*
6	NNCSSS	_ _ C B A D	Acc += C * w3	*C recircula*
7	NLCSSS	_ E D C B A	E = Acc += D * w4	*D recircula y E se carga*
	. . . . .	. . . . .	.....	
11	SLSSSS	_ F E D C B	F = Acc += A * w4	*desplazamiento y F se carga*
12	SSSSSS	_ _ F E D C	cálculo de G	*desplazamiento global*
	. . . . .	. . . . .	.....	

El microprograma coloca en la fila de datos los valores resultantes de calcular cada neurona, en una posición tal, que dicho valor opere con el peso adecuado al momento de salir de la fila. Para esto, los pesos se almacenan en un buffer circular en memoria ROM.

De esta manera, la RND se comporta como un sistema sistólico que en cada ciclo de reloj multiplica un valor proveniente de la fila de datos/resultados por su peso sináptico correspondiente acumulando el resultado parcial hasta procesar todas las entradas de cada neurona. Una vez obtenida la salida de la neurona procesada, se le aplica la función sigmoide y se deposita el resultado dentro de la fila de datos, en la celda habilitada por la microinstrucción actual.

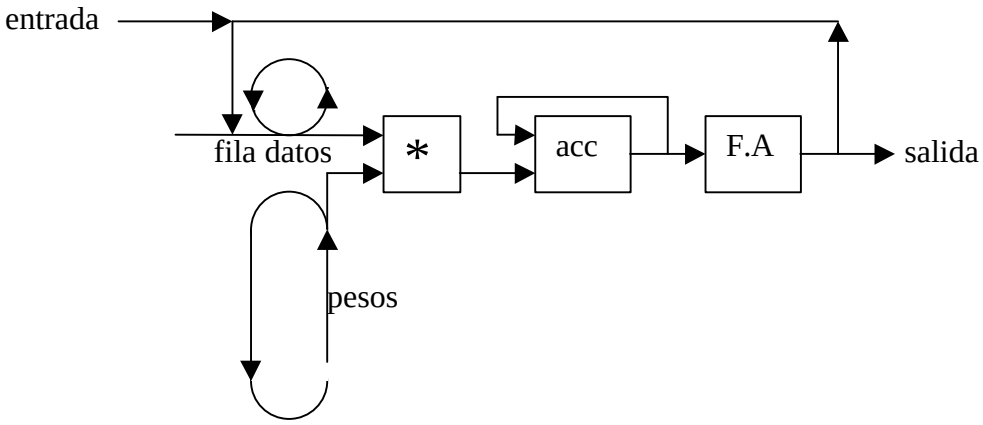


Figura 6: flujo de datos de la RND

4. Resultados en simulación

Implementación del modelo

Se implementó el modelo descripto de manera que se pudieran analizar series temporales variando todos los parámetros de la red: número de series de entrada, cantidad de muestras pasadas de cada serie de entrada (n_k), cantidad de muestras pasadas de la salida realimentada (n_i), número de neuronas ocultas, parámetros de aprendizaje, etc.. La figura 7 muestra una comparación de una serie de salida de la red neuronal junto con la serie real con que fue entrenada, usando una serie temporal

de entrada de 250 muestras, una serie de salida de 250 muestras y una red neuronal de 4 entradas (n_i=2, n_k=2) con 5 neuronas en su capa oculta.

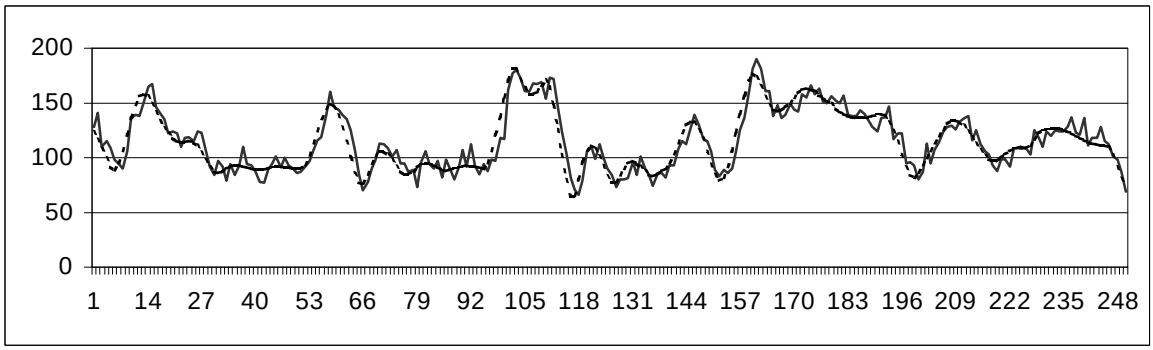


Figura 7: Salida de la red neuronal (puntos) vs. serie real de entrenamiento (trazo continuo).

Análisis de precisión

Cualquier implementación digital presenta un compromiso de diseño entre precisión, por un lado, y parámetros físicos como área y velocidad de cálculo por el otro. Es necesario, entonces, analizar el comportamiento de la red neuronal entrenándola con diferentes series de datos y con distintas precisiones internas (número de bits de los valores).
Para la aplicación objeto de este diseño se optó por una precisión de 8 bits para los datos de entrada y salida de la red así como para los pesos sinápticos basándose en dos criterios. En primer lugar, los datos de entrada de la red provienen de sensores (temperatura, humedad) que, por lo general, entregan un valor analógico 'discretizado' a valores digitales en rangos de 0..255 o cualquier otro rango desplazado según algún parámetro de referencia (-127..128; -63..196; etc.), representables con 8 bits. Por otro lado, se analizó el comportamiento de la red a través de sucesivas simulaciones de la misma con distintas situaciones de entrenamiento usando representaciones para los datos y pesos de 20, 16, 12 y 8 bits. Los resultados obtenidos para el último caso –8 bits- ofrecen un margen de error aceptable para la aplicación en la que la red neuronal va a ser usada, esto es, predicción de estados en 1 tiempo futuro para un entorno con tiempo de respuesta –acción/reacción- lento.
La figura 8 muestra una comparación entre la salida de la red neuronal para las series correspondientes a la figura 7 y la salida del modelo simulado con precisión de 8 bits.

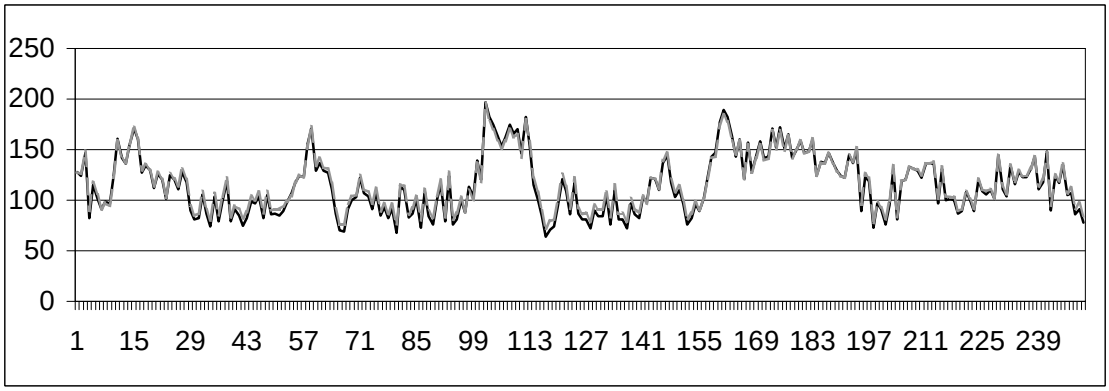


Figura 8: Salida de la red con precisión real (trazo de rayas) vs salida con precisión de 8 bits (trazo continuo)

Todos los componentes citados se implementan como módulos parametrizables en VHDL (Vhsic Hardware Description Language) de manera que se puedan generar en diferentes precisiones (ancho de cálculo). Para el caso de la memoria de pesos sinápticos y la fila de datos/resultados, su descripción VHDL se genera a partir de parámetros surgidos de la topología específica de la RND.

5. Conclusiones

Si bien el proyecto global, en que se incluye este trabajo no está aún finalizado, y por consiguiente, no se han realizado pruebas exhaustivas de campo, se pudo experimentar la arquitectura presentada en distintos estadios de su desarrollo. Se dedicó tiempo y esfuerzos al análisis de la mecánica de funcionamiento de las RN aplicadas a la predicción, así como a la definición y prueba de cada uno de los componentes básicos, tanto a nivel teórico como en su implementación y simulación lógica.
Por otro lado, se logró cumplir con uno de los objetivos iniciales referido a posibilidad de sintetizar el diseño completo de la RNA en una FPGA de tamaño moderado como lo es la serie XC4010D de la familia XILINX con la que se trabajó.
Las tareas a futuro se relacionan con la redefinición de la unidad de control a fin de optimizar las secuencias de microinstrucciones y el análisis de solapamiento de cálculo de sucesivas decisiones a fin de minimizar los ciclos ociosos de la ruta de datos debidos a los períodos de llenado y vaciado de las etapas de pipeline. Asimismo, se espera poder comprobar la arquitectura propuesta en base a resultados experimentales sobre el prototipo descripto.

Referencias

- Acosta, Gerardo, Alonso González, Carlos, Spina, Marcelo, de la Vega, Roberto, (2000), “On the application of a task taxonomy for Knowledge Bases Supervision in Climate Control of a greenhouse”, Aceptado para su publicación, Proc. of 4th IFAC International Symposium on Intelligent Components and Instruments for Control Applications SICICA 2000, Sept. 13 al 15, Buenos Aires.

- Koskela, T. Lehtokangas, J. Saarinen and K. Kaski, (1996), “Time Series Prediction With Multilayer Perceptron, FIR and Elman Neural Networks”, Proceedings of the World Congress on Neural Networks, INNS Press, San Diego, USA, pp 491-496, 1996.

- Corchado J., Fyfe C. y Lees B. , (1998), “Unsupervised Neural Method for Temperature Forecasting”, Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems EIS´98, Volume 2, Neural Networks, pp 300-306, 11 al 13 de Febrero de 1998.

- Hagan, Martin T. y Menhaj, Mohamed B., (1994), “Training Feedforward Networks with the Marquardt Algorithm”, IEEE Transactions on Neural Networks, Vol. 5, No. 6, November 1994.

- Tosini, Marcelo, (2000), “Implementación en Harware de una Ruta de datos segmentada para Redes Neuronales”, VI Workshop Iberchip, IWS´2000, Sao Paulo, Brasil, pp 292-299, 16 al 18 de Marzo de 2000.

- Nordström, T. and Svensson, B., 'Using and Designing Massively Parallel Computers for Artificial Neural Networks', Journal of Parallel and Distributed Processing, vol. 14, no. 3, pp. 260-285, march 1992

- Gurney, Kevin, “Computers and Symbols Versus Nets and Neurons”. UCL Press Limited, UK., 1995.

- Myers, D. J. and Hutchinson, R. A., (1989), “Efficient implementation of piecewise linear activation function for digital VLSI neural networks”, Electronic Letters, Vol. 25, pp 1662-1663, 1989.

- Alippi, C. and Storti-Gajani, G.,”Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning”, Proc. IEEE Int. Symp. Circuits and Systems, pp 1505-1508, 1991.